
Pengenalan Sains Komputasi - Movement of Ants

Ridlo W. Wibowo || 20912009

December 11, 2012

Problem.
Apply the model of ant movement in a 17 $\times$ 17 grid where there are 20 ant's initially located randomly. Assume that there is food right in the middle of the grid, in which the initial amount of chemical is 10, where in the other cells, the amount is 0. Run the program 5 times, each for t=10, t=20, ... , t=100, and write your conclusion of the simulation.

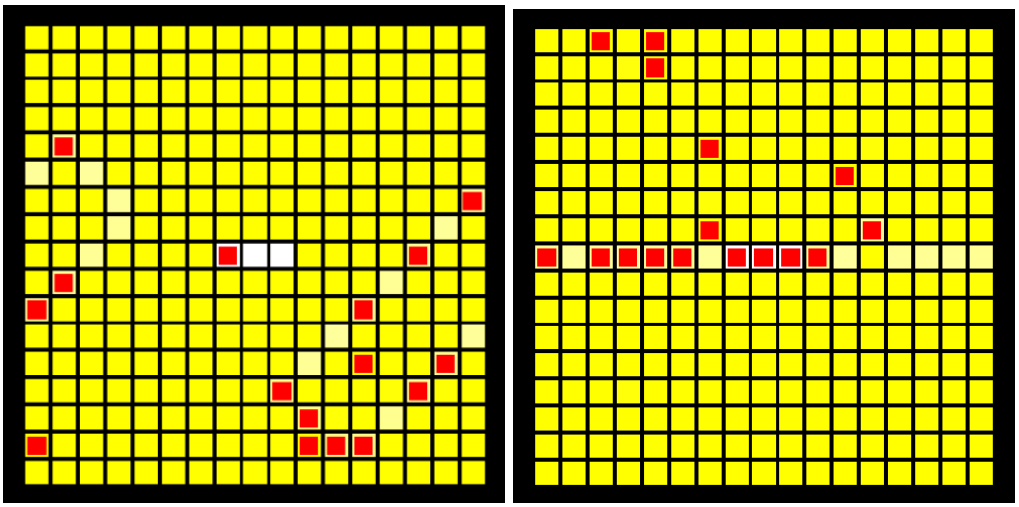
Documentation.
Dengan hanya menggunakan asumsi adanya pheromon sebagai 'petunjuk arah' bagi semut, dan apabila yang dicek adalah untuk semua arah, maka yang akan terjadi adalah semut akan bolak-balik di tempat yang sama. Oleh karena itu dilakukan sedikit perubahan dalam membuat simulasi ini, yaitu:

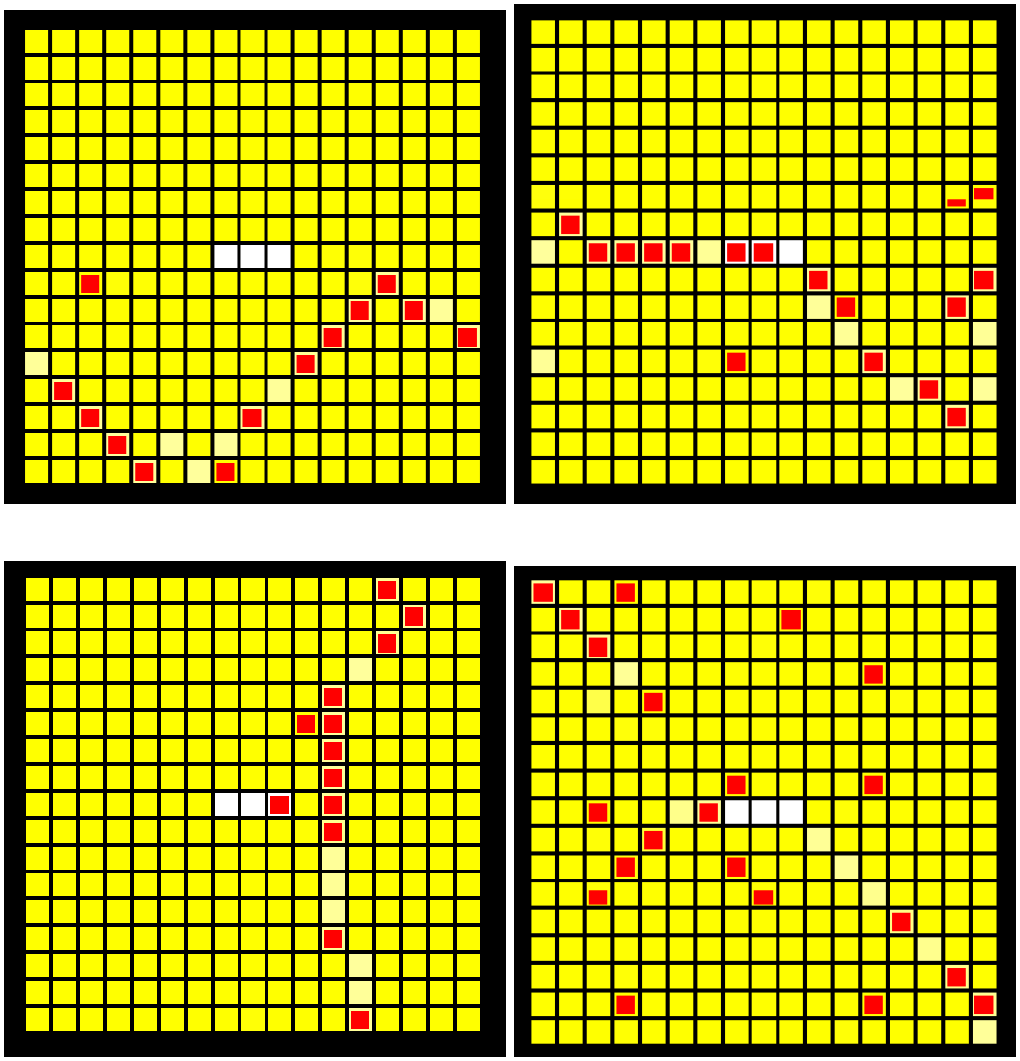
- simulasi melibatkan 8 arah.
- semut hanya memeriksa keberadaan pheromon di 5 grid di arah dimana ia menghadap (depan dan samping) dari 8 grid tersebut.
- semut sebagai objek (bukan cellular automaton), sehingga diperbolehkan semut berada pada grid yang sama, dan pada simulasi kali ini diasumsikan tidak terjadi tumbukan.
- periodic boundary
- ada 3 makanan di tengah (dengan menganggap pheromon tetap selama iterasi)

- terdapat *probabilitas semut bosan* untuk mengurangi peluang semut berputar pada kumpulan grid dan memperbesar peluang semut sampai di makanan (dalam simulasi ini saya ambil 0.1).
- penambahan pheromon setelah grid ditempati semut dan pengurangan pheromon setiap waktu dilakukan secara linear menggunakan bilangan bulat saja.

Menggunakan beberapa asumsi di atas lalu dibuat simulasi sederhana tersebut, menggunakan *C++* dan *OpenGL*. Warna kuning menunjukkan grid tempat semut, semakin putih semakin banyak pheromonnya. Tiga grid putih ditengah merupakan makanan, dan semut disimbolkan dengan grid merah. Karena keterbatasan waktu kode masih kurang terstruktur dengan baik, dan banyak penggunaan *IF* yang seharusnya dapat dihindari (program terlampir).

Screenshot setelah cukup stabil:





Screenshot kondisi setelah cukup stabil. Kotak kuning sebagai grid, semakin putih warnanya semakin banyak jumlah pheromon, tiga kotak putih ditengah merupakan grid yang dibuat konstan jumlah pheromonnya, sedangkan gird merah melambangkan posisi semut.

cuplikan animasi dapat dilihat di `http://www.youtube.com/watch?v=DWg_oeGBZp4/`

Diskusi

1. semakin lama waktu iterasi maka semut akan cenderung berkumpul atau membuat suatu jalur.
2. untuk membuat simulasi lebih real sepertinya dapat dilakukan dengan menggunakan dua tipe pheromon yakni pheromon makanan dan pheromon rumah, dan semut berasal dari suatu tempat yang sama (koloni).

3. dalam kondisi nyata memang cenderung tidak ada semut yang balik arah.

4. perlu pemodelan penambahan dan pengurangan jumlah pheromon yang lebih baik (dalam program ini pheromon hanya berupa bilangan integer, dan bertambah/berkurang secara linear saja).

Program

```
1  // Copyleft (c) Ridlo W. Wibowo
2  // Simut
3  #include <iostream>
4  #include <stdlib.h>
5  #include <GL/gl.h>
6  #include <GL/glut.h>
7  #include <fstream>
8  #include <time.h>
9  #define MAX 17
10 #define maxSemut 20
11 using namespace std;
12
13 int grid [MAX][MAX];
14 int grid2[MAX][MAX];
15 int strong[MAX][MAX];
16 bool f = false;
17 int tpm = 100;
18 double boredProb = 0.1;
19 int maxPher = 200;
20 /* N  = 0, NE = 1, E  = 2, SE = 3, S  = 4, SW = 5, W  = 6, NW = 7 */
21
22 double unirand(){ return (double)rand()/(double)RAND_MAX;}
23
24 int getStrong(int x, int y, int direc){
25     int m = x+MAX; int n = y+MAX; int kuat=0; int arahbaru = direc;
26     if (direc == 0){
27             if (grid[m%MAX][(n+1)%MAX] > kuat){kuat = grid[m%MAX][(n+1)%MAX
                   ]; arahbaru=0;}
28             if (grid[(m-1)%MAX][n%MAX] > kuat){ kuat = grid[(m-1)%MAX][n%
                   MAX] ; arahbaru = 6;}
29             if (grid[(m-1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                   ][(n+1)%MAX] ; arahbaru = 7;}
30             if (grid[(m+1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                   ][(n+1)%MAX] ; arahbaru = 1;}
31             if (grid[(m+1)%MAX][n%MAX] > kuat){ kuat = grid[(m+1)%MAX][n%
                   MAX] ; arahbaru = 2;}
32     }
33     if (direc == 1){
34             if (grid[m%MAX][(n+1)%MAX] > kuat){kuat = grid[m%MAX][(n+1)%MAX
                   ]; arahbaru = 0;}
35             if (grid[(m+1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                   ][(n+1)%MAX] ; arahbaru = 1;}
```

```
36              if (grid[(m-1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n+1)%MAX] ; arahbaru = 7;}
37              if (grid[(m+1)%MAX][n%MAX] > kuat){ kuat = grid[(m+1)%MAX][n%
                    MAX] ; arahbaru = 2;}
38              if (grid[(m+1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 3;}
39      }
40      if (direc == 2){
41              if (grid[m%MAX][(n+1)%MAX] > kuat){kuat = grid[m%MAX][(n+1)%MAX
                    ]; arahbaru = 0;}
42              if (grid[(m+1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n+1)%MAX] ; arahbaru = 1;}
43              if (grid[(m+1)%MAX][n%MAX] > kuat){ kuat = grid[(m+1)%MAX][n%
                    MAX] ; arahbaru = 2;}
44              if (grid[(m+1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 3;}
45              if (grid[m%MAX][(n-1)%MAX] > kuat){ kuat = grid[m%MAX][(n-1)%
                    MAX] ; arahbaru = 4;}
46      }
47      if (direc == 3){
48              if (grid[(m+1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n+1)%MAX] ; arahbaru = 1;}
49              if (grid[(m+1)%MAX][n%MAX] > kuat){ kuat = grid[(m+1)%MAX][n%
                    MAX] ; arahbaru = 2;}
50              if (grid[(m+1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 3;}
51              if (grid[m%MAX][(n-1)%MAX] > kuat){ kuat = grid[m%MAX][(n-1)%
                    MAX] ; arahbaru = 4;}
52              if (grid[(m-1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 5;}
53      }
54      if (direc == 4){
55              if (grid[(m+1)%MAX][n%MAX] > kuat){ kuat = grid[(m+1)%MAX][n%
                    MAX] ; arahbaru = 2;}
56              if (grid[(m+1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 3;}
57              if (grid[m%MAX][(n-1)%MAX] > kuat){ kuat = grid[m%MAX][(n-1)%
                    MAX] ; arahbaru = 4;}
58              if (grid[(m-1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 5;}
59              if (grid[(m-1)%MAX][n%MAX] > kuat){ kuat = grid[(m-1)%MAX][n%
                    MAX] ; arahbaru = 6;}
60      }
61      if (direc == 5){
62              if (grid[(m+1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 3;}
63              if (grid[m%MAX][(n-1)%MAX] > kuat){ kuat = grid[m%MAX][(n-1)%
                    MAX] ; arahbaru = 4;}
64              if (grid[(m-1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 5;}
65              if (grid[(m-1)%MAX][n%MAX] > kuat){ kuat = grid[(m-1)%MAX][n%
                    MAX] ; arahbaru = 6;}
66              if (grid[(m-1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n+1)%MAX] ; arahbaru = 7;}
```

```
67      }
68      if (direc == 6){
69              if (grid[m%MAX][(n-1)%MAX] > kuat){ kuat = grid[m%MAX][(n-1)%
                    MAX] ; arahbaru = 4;}
70              if (grid[(m-1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 5;}
71              if (grid[(m-1)%MAX][n%MAX] > kuat){ kuat = grid[(m-1)%MAX][n%
                    MAX] ; arahbaru = 6;}
72              if (grid[(m-1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n+1)%MAX] ; arahbaru = 7;}
73              if (grid[m%MAX][(n+1)%MAX] > kuat){kuat = grid[m%MAX][(n+1)%MAX
                    ]; arahbaru = 0;}
74      }
75      if (direc == 7){
76              if (grid[(m-1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n-1)%MAX] ; arahbaru = 5;}
77              if (grid[(m-1)%MAX][n%MAX] > kuat){ kuat = grid[(m-1)%MAX][n%
                    MAX] ; arahbaru = 6;}
78              if (grid[(m-1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n+1)%MAX] ; arahbaru = 7;}
79              if (grid[m%MAX][(n+1)%MAX] > kuat){kuat = grid[m%MAX][(n+1)%MAX
                    ]; arahbaru = 0;}
80              if (grid[(m+1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n+1)%MAX] ; arahbaru = 1;}
81      }
82
83      return arahbaru;
84      /*
85      for (int i=0;i<MAX;i++){
86          for (int j=0;j<MAX;j++){
87              int m = i + MAX; int n = j + MAX; int kuat=0; //strong[i][j] =
                    0;
88              if (grid[m%MAX][(n+1)%MAX] > kuat){kuat = grid[m%MAX][(n+1)%MAX
                    ]; strong[i][j] = 0;}
89              if (grid[(m+1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n+1)%MAX] ; strong[i][j] = 1;}
90              if (grid[(m+1)%MAX][n%MAX] > kuat){ kuat = grid[(m+1)%MAX][n%
                    MAX] ; strong[i][j] = 2;}
91              if (grid[(m+1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m+1)%MAX
                    ][(n-1)%MAX] ; strong[i][j] = 3;}
92              if (grid[m%MAX][(n-1)%MAX] > kuat){ kuat = grid[m%MAX][(n-1)%
                    MAX] ; strong[i][j] = 4;}
93              if (grid[(m-1)%MAX][(n-1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n-1)%MAX] ; strong[i][j] = 5;}
94              if (grid[(m-1)%MAX][n%MAX] > kuat){ kuat = grid[(m-1)%MAX][n%
                    MAX] ; strong[i][j] = 6;}
95              if (grid[(m-1)%MAX][(n+1)%MAX] > kuat){ kuat = grid[(m-1)%MAX
                    ][(n+1)%MAX] ; strong[i][j] = 7;}
96          }
97      }
98      */
99
100 }
101
```

```
102 class varSemut{
103     public:
104         int x, y, arah;
105         void set_value(int, int, int);
106         void sensing();
107         void buangpher();
108 } semut[maxSemut];
109
110 void varSemut::set_value(int a, int b, int c){x = (a+MAX)%MAX; y = (b+MAX)%
    MAX; arah = c;}
111
112 void varSemut::buangpher(){
113     if (grid2[x][y] < 50){
114         grid2[x][y] = grid2[x][y] + 2;
115     }
116 }
117
118 void varSemut::sensing(){
119     int arahnew = getStrong(x, y, arah);
120     if (arahnew == 0){
121         x = x;
122         y = (y+MAX+1)%MAX;
123         arah = arahnew;
124     }
125     if (arahnew == 1){
126         x = (x+MAX+1)%MAX;
127         y = (y+MAX+1)%MAX;
128         arah = arahnew;
129     }
130     if (arahnew == 2){
131         x = (x+MAX+1)%MAX;
132         y = y;
133         arah = arahnew;
134     }
135     if (arahnew == 3){
136         x = (x+MAX+1)%MAX;
137         y = (y+MAX-1)%MAX;
138         arah = arahnew;
139     }
140     if (arahnew == 4){
141         x = x;
142         y = (y+MAX-1)%MAX;
143         arah = arahnew;
144     }
145     if (arahnew == 5){
146         x = (x+MAX-1)%MAX;
147         y = (y+MAX-1)%MAX;
148         arah = arahnew;
149     }
150     if (arahnew == 6){
151         x = (x+MAX-1)%MAX;
152         y = y;
153         arah = arahnew;
154     }
```

```
155     if (arahnew == 7){
156         x = (x+MAX-1)%MAX;
157         y = (y+MAX+1)%MAX;
158         arah = arahnew;
159     }
160 }
161
162 void inisiasi();
163
164 void menu(int t){
165   tpm = t;
166   glutPostRedisplay();
167 }
168
169 void copy(){
170   for(int i = 0; i < MAX; i++){
171     for(int j = 0; j < MAX; j++){
172       grid[i][j] = grid2[i][j];}}
173 }
174
175 void mlaku(){
176     for (int k=0;k<maxSemut;k++){
177         if (unirand() > boredProb){
178             semut[k].buangpher();
179             semut[k].sensing();
180         }
181         else {
182             semut[k].buangpher();
183             semut[k].set_value(semut[k].x + (-1 + (rand()%3)), semut[k].y +
                    (-1 + (rand()%3)), rand()%8); //semutnya bosen
184         }
185     }
186 }
187
188 void jalan(){
189     for (int i=0;i<MAX;i++){
190         for (int j=0;j<MAX;j++){
191             if (grid[i][j] > 0){ grid2[i][j] -= 1;}
192         }
193     }
194     grid2[8][8] = 100;
195     grid2[8][7] = 100;
196     grid2[8][9] = 100;
197     //grid2[7][8] = 100;
198     //grid2[9][8] = 100;
199     mlaku();
200     copy();
201 }
202
203 void par(float x1, float x2, float y1, float y2, int val){
204     glColor3f(1.0, 1.0, (float)val/(float)maxPher);
205   glBegin(GL_QUADS);
206   glVertex3f(x1, y1, 0.0);
207   glVertex3f(x2, y1, 0.0);
```

```
208    glVertex3f(x2, y2, 0.0);
209    glVertex3f(x1, y2, 0.0);
210    glEnd();
211  }
212
213  void sem(float x1, float x2, float y1, float y2){
214      glColor3f(1.0, 0.0, 0.0);
215    glBegin(GL_QUADS);
216    glVertex3f(x1, y1, 0.0);
217    glVertex3f(x2, y1, 0.0);
218    glVertex3f(x2, y2, 0.0);
219    glVertex3f(x1, y2, 0.0);
220    glEnd();
221  }
222
223  void display(void){
224    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
225    glMatrixMode(GL_MODELVIEW);
226    glLoadIdentity();
227    glTranslatef(0.0, 0.0, -22.0);
228      for (int k=0;k<maxSemut;k++){
229          sem(-6.0 + 0.7 * semut[k].y + 0.15,
230              -6.0 + 0.7 * (semut[k].y + 1) - 0.075,
231               6.0 - 0.7 * semut[k].x + 0.15,
232               6.0 - 0.7 * (semut[k].x - 1) - 0.075);
233      }
234    for(int i = 0; i < MAX; i++){
235      for(int j = 0; j < MAX; j++){
236          par(-6.0 + 0.7 * j + 0.1,
237            -6.0 + 0.7 * (j + 1),
238            6.0 - 0.7 * i + 0.1,
239            6.0 - 0.7 * (i - 1),
240                      grid[i][j]);
241      }
242    }
243    glutSwapBuffers();
244  }
245
246  void myIdleFunc(int a) {
247    jalan();
248    glutPostRedisplay();
249    if(f) glutTimerFunc(tpm, myIdleFunc, 0);
250  }
251
252  void inisiasi(){
253    for (int i=0;i<MAX;i++){
254          for (int j=0;j<MAX;j++){
255              grid2[i][j] = 0;
256          }
257      }
258      grid2[8][8] = 100;
259      grid2[8][7] = 100;
260      grid2[8][9] = 100;
261      //grid2[7][8] = 100;
```

```
262     //grid2[9][8] = 100;
263     copy();
264     for (int k=0;k<maxSemut;k++){
265         semut[k].set_value(rand()%MAX, rand()%MAX, rand()%8);
266     }
267 }
268
269 void keyboard(unsigned char key, int x, int y){
270   if(key == 27) {
271     exit(0);
272   }else if((char)key == 'a'){
273     if(!f) glutTimerFunc(tpm, myIdleFunc, 0);
274     f = true;
275   }else if((char)key == 's'){
276     jalan();
277     glutPostRedisplay();
278   }else if((char)key == 'd'){
279     f = false;
280   }else if((char)key == 'f'){
281     inisiasi();
282         f = false;
283     glutPostRedisplay();
284     }
285 }
286
287 void init(){
288   glEnable(GL_DEPTH_TEST);
289   glEnable(GL_COLOR_MATERIAL);
290   glEnable(GL_LIGHTING);
291   glEnable(GL_LIGHT0);
292   glEnable(GL_NORMALIZE);
293   glShadeModel(GL_SMOOTH);
294   glLoadIdentity ();
295   glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
296   GLfloat acolor[] = {1.4, 1.4, 1.4, 1.0};
297   glLightModelfv(GL_LIGHT_MODEL_AMBIENT, acolor);
298 }
299
300 void Reshape(int w, int h){
301     glViewport(0, 0, w, h);
302     glMatrixMode(GL_PROJECTION);
303   glLoadIdentity();
304   gluPerspective(45.0, (float)w/(float)h, 0.1, 200.0);
305 }
306
307 int main(int argc, char** argv){
308   srand(time(NULL));
309     inisiasi();
310   glutInit(&argc,argv);
311   glutInitDisplayMode ( GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
312   glutInitWindowSize(700,700);
313   glutInitWindowPosition(500,0);
314   glutCreateWindow("Semut muutmuut");
315   glutCreateMenu(menu);
```

```
316     glutAddMenuEntry( "20",  20);
317     glutAddMenuEntry( "40",  40);
318     glutAddMenuEntry( "60",  60);
319     glutAddMenuEntry("100", 100);
320     glutAddMenuEntry("150", 150);
321     glutAddMenuEntry("200", 200);
322     glutAttachMenu(GLUT_RIGHT_BUTTON);
323     init();
324     glutReshapeFunc(Reshape);
325     glutKeyboardFunc(keyboard);
326     glutDisplayFunc(display);
327     glutMainLoop();
328     return 0;
329 }
```